REMARKS

Claims 1 to 28 are pending in this application. Of these, claims 1 and 14 are independent.[1] Favorable reconsideration and further examination are respectfully requested.

All of the claims were rejected over U.S. Patent Publication No. 2002/0168621 (Cook) in view of U.S. Patent Publication No. 2006/0059253 (Goodman). Applicants respectfully traverse this rejection for at least the following reasons.

Independent claim 1 recites a method, performed by one or more processing devices, for use in an electronic learning system that stores information as learning objects. The method comprises designating a target learning object as a project object, and storing version dependency data in the project object. The version dependency data identifies at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly depends.

As explained in the Amendment dated December 5, 2006, Cook fails to disclose or to suggest the features of claim 1, in particular, storing version dependency data in the project object, where the version dependency data identifies at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly depends. Although page 3 of the Office Action seems to indicate that paragraphs 0078 to 0081 show all features of claim 1[2], the following sentence appears to agree

[1] The Examiner is urged to independently confirm this recitation of the pending claims.
[2] Office Action, page 3, the paragraph beginning "Cook discloses a method,...."

with Applicants' argument: "Cook does not explicitly disclose the version dependency data as claimed"[3]. Goodman was cited to make up for this deficiency of Cook.

Goodman describes a system that keeps track of different versions of software, e.g., through version control services, as described below.

> As with standard development code, when media content is created and edited, the version control services maintain a revision history of changes. This way, if it is necessary to revert to an original piece of media content, it is not necessary to go all the way back to the original source (which in the case of finding an image in a CD-ROM library containing 10,000 images, for example, could be a difficult task). In practice, this may mean storing the original and final copies of media (especially where volume is an issue). For this reason, a process for managing multiple versions of media content is put into place in the preferred development architecture 500.[4]

Paragraph 0234 describes version control as follows:

> Version control and compatibility are key considerations when managing these packages. Note that version control not only applies to software components, but also to all components of a given package, including test scripts, test data and design documentation. It is also of great importance to keep track of which version is in which environment. If incompatibilities are discovered, it must always be possible to "roll back" to a previous consistent state; that is, to revert to an earlier version of one or more components. It must be possible to define releases of a configuration--a list of version numbers, one for each component of the package, which, together, form a consistent configuration. The smallest unit that can be version-controlled should be the package as defined in the packaging plan. This ensures that the lowest common denominator in all version-control activities is managed at the package level.

Among ways that Goodman enables access to different versions, is by providing a different – called "shadow" – repository for prior object versions.[5]

Thus, Goodman does describe maintaining and accessing different versions of software objects. The Office Action cites paragraphs 0138, 0170 and 0173 for this proposition.[6] The

---

[3] Office Action, page 3
[4] Paragraph 0176
[5] Paragraph 0307
[6] Office Action, page 3: "Goodman furthermore teaches version control in paragraphs 0138, 0170, and 0173."

Office Action further states that "Goodman teaches the object and data dependencies in paragraphs 0316 and 0378". These paragraphs are reproduced below for discussion purposes.

> [0316] As illustrated in FIG. 26, the information management tools 602 is also responsible for object management 646. Object management tools provide capabilities for viewing objects, their methods and attributes, and the dependencies between these objects. Object management tools also provide specific analysis tools, in order to understand interdependencies between the core classes and the components. When classes and components are modified, impact analysis tools are required to see where the modified entity is being used, allowing them to understand what is the overall impact of the change. This is more complex than with traditional systems as a veritable spider's web of dependencies between classes, components, and applications may ensue. In addition, OO features such as inheritance and polymorphism make tracking down dependencies with simple text search tools much more difficult.
>
> [0378] Process modeling tools 600 provide a graphical depiction of the business functions and processes being supported by a system. The tool(s) selected must support the modeling techniques being used in the development methodology; these include process decomposition, data flow and process dependency. Process modeling is a method for clarifying and communicating the business design of the system. The process model can provide an effective means of bringing people together, creating a shared vision of how the business is to function. A process model provides a means of standardizing a set of similar processes, which exist, for example, at different branches of the business.

Paragraph 0316 does describe dependencies among objects, and "specific analysis tools...to understand interdependencies between the core classes and the components". Paragraph 0378, on the other hand, merely mentions "process dependency". Significantly, however, neither these paragraphs, nor the remainder of Goodman, describes exactly how the dependencies are analyzed or managed in connection with different versions. Simply reciting version control and the existence of dependencies is clearly not enough to render obvious claim 1's very specific method, which includes storing version dependency data in a project object, where the version dependency data identifies at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly

depends. For example, Goodman does not provide any hint of how indirect dependencies are managed, much less that they are stored in the same project object as direct dependencies.

For at least the foregoing reasons, claim 1 is believed to be patentable over Cook and Goodman, whether taken alone or in combination. Independent claim 14 is a program claim that roughly corresponds to claim 1, and is also believed to be patentable.

Regarding dependent claims 27 and 28, there is nothing in Cook or Goodman to disclose or suggest that a distinction between object dependency data and version dependency data, as claimed. In this regard, the Office Action merely cites paragraph 0316 of Goodman and restates the language of claims 27 and 28, without any explanation.[7]

> [0316] As illustrated in FIG. 26, the information management tools 602 is also responsible for object management 646. Object management tools provide capabilities for viewing objects, their methods and attributes, and the dependencies between these objects. Object management tools also provide specific analysis tools, in order to understand interdependencies between the core classes and the components. When classes and components are modified, impact analysis tools are required to see where the modified entity is being used, allowing them to understand what is the overall impact of the change. This is more complex than with traditional systems as a veritable spider's web of dependencies between classes, components, and applications may ensue. In addition, OO features such as inheritance and polymorphism make tracking down dependencies with simple text search tools much more difficult.

As set forth above, paragraph 0316 does mention dependencies among objects, but says nothing whatsoever about different object dependency data and different version dependency data, much less that the version of the first object and the version of the second object store object dependency data but not version dependency data, the object dependency data for the version of the first object identifies one or more first learning objects upon which the version of the first object depends but does not identify versions of the one or more first learning objects, and the

[7] Office Action, pages 12 and 13

object dependency data for the version of the second object identifies one or more second learning objects upon which the version of the second object depends but does not identify versions of the one or more second learning objects. For at least this reason, claims 27 and 28 are believed to be patentable over Cook and Goodman.

The remaining dependent claims are also believed to define patentable features of the invention. Each dependent claim partakes of the novelty of its corresponding independent claim and, as such, each has not been discussed specifically herein.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

In view of the foregoing amendments and remarks, Applicants respectfully submit that the application is in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicants' undersigned attorney can be reached at the address shown below. All telephone calls should be directed to the undersigned at 617-521-7896.

Please apply any fees or credits due in this case, which are not already covered by check, to Deposit Account 06-1050 referencing Attorney Docket No. 13909-161001.

Respectfully submitted,

Date: May 9, 2007

Paul A. Pysher
Reg. No. 40,780

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

161.doc